Product RAMP Requirements

DOCUMENT IDENTIFIER: A-DS-ELX0091-00-0000 Rev A, 10-Jun-1991

ABSTRACT: This document defines a set of Reliability, Availability, Maintainability Program (RAMP) requirements to be used by cross-functional hardware and software product teams in the selection of the product attributes needed to support both customer requirements and service business objectives and strategies. This standard presents a minimum set of product RAMP and software/firmware support requirements to be considered: it is not intended to be all-inclusive. This document is an external version of DEC STD 091-0, Rev A.

APPLICABILITY: This document is mandatory for all new hardware products developed or purchased by Digital that will be serviced by Digital Services. It is also mandatory for all Software and firmware products that provide support for the new hardware products.

© Digital Equipment Corporation. 1991. All rights reserved.

Product RAMP Requirements

DOCUMENT IDENTIFIER: A-DS-ELX0091-00-0000 Rev A, 10-Jun-1991

REVISION HISTORY:	No previous revision exists.
Document Management Category:	Customer Services Requirements (FR)
Responsible Department:	Customer Service Systems Engineering
Responsible Person:	Bill Bazemore

Bill Bazemore - Customer Service Systems Engineering

CONTENTS

1 INTRODUCTION 1
1.1 PURPOSE 1
1.2 SCOPE 1
1.3 RESPONSIBILITIES 2
1.3.1 Engineering Development Manager 2
1.3.2 Product Team 2
1.3.3 Services Representative 2
1.3.4 Advisory Committee 2
1.4 RAMP GOALS 2

2 HARDWARE PRODUCT RAMP REQUIREMENTS 4
2.1 RELIABILITY REQUIREMENTS 4
2.1.1 Data Integrity 4
2.1.2 Electrical Design Integrity 5
2.1.3 Thermal Design Integrity 5
2.1.4 Environmental Sensing and Shutdown Protection 5
2.2 AVAILABILITY REQUIREMENTS 5
2.3 MAINTAINABILITY REQUIREMENTS 6
2.3.1 Product Requirements 6
2.3.1.1 Error Detection 7
2.3.1.2 Error Information Capture and Reporting 7
2.3.1.3 Fault Isolation 8
2.3.1.4 On-site Repair Process—Physical Repair 8
2.3.1.5 Verify/Restore 10
2.3.1.6 Revision and Control 10
2.3.1.7 Safety 10
2.3.2 Diagnosis Requirements 10
2.3.2.1 Symptom-Directed Diagnosis Requirements 11
2.3.2.2 Test-Directed Diagnosis Requirements 11
2.3.2.3 Exerciser Program Requirements 13
2.3.2.4 Utility Program Requirements 13
2.3.2.5 Verification of Diagnostic Tools 13
2.3.3 Training Requirements 13
2.3.4 Documentation Requirements 14
2.4 INSTALLATION REQUIREMENTS 14

3 SOFTWARE AND FIRMWARE SUPPORT OF HARDWARE RAMP REQUIREMENTS 14
3.1 RELIABILITY REQUIREMENTS 15
3.2 AVAILABILITY REQUIREMENTS 15
3.2.1 Configuration Monitoring 15
3.2.2 Product Error Handling and Reporting 15
3.2.3 Fault Tolerance Support 15

3.3 MAINTAINABILITY (SERVICEABILITY) REQUIREMENTS 16
3.3.1 Diagnostic Support 16
3.3.2 Event/Error Logging Support 16
3.3.3 Revision Management 17
3.4 INSTALLATION REQUIREMENTS 17
3.4.1 Software Installation Procedures 17
3.4.2 Verification Tools 17
3.5 SOFTWARE PRODUCT DOCUMENTATION 18
3.5.1 Software Installation Manual 18
3.5.2 Software Documentation 18

Appendix A EXPANDED DEFINITION OF RAMP GOALS 19

GLOSSARY 25

1 INTRODUCTION

Digital's long-term strategy is to provide total support for our customers throughout the life cycle of our products by providing products and services that support the customer's style of computing. The key to a customer's purchase decisions is Digital's ability to provide the right solution, competitive cost of ownership, uninterrupted applications, and high system availability.

A key to implementing this strategy is to provide a consistent high level of product reliability, availability, and maintainability across all product offerings. This strategy can be accomplished by the establishment of a clear set of product Reliability, Availability, Maintainability Program (RAMP) goals and requirements that can be used by each product team in establishing its product requirements. This shall be done through an integrated product team effort by Engineering, Product Management, Marketing, Sales, Manufacturing, and Services.

This standard is divided into three major headings:

1. Heading 1 contains the introduction and the required product RAMP goals.
2. Heading 2 defines the RAMP hardware product requirements that shall be addressed by each hardware product development team.
3. Heading 3 defines the software and firmware requirements that shall be addressed by all development teams designing products that provide support for new hardware products. This support may include any or all of the following products: networks, systems, subsystems, operating systems, device drivers, layered products, applications, or any other facilities that might support hardware products.

Headings 2 and 3 (Hardware and Software/Firmware, respectively) follow the same RAMP format, address similar related requirements, and contain some duplicate information for ease of use by the intended audiences.

1.1 PURPOSE

This standard defines a set of hardware, firmware, and software product RAMP requirements to be used in the selection of product attributes needed to support both customer requirements and service business objectives.

1.2 SCOPE

This standard applies to all new hardware products developed or purchased by Digital that will be serviced by Digital Services. It also applies to all software and firmware products that provide support for new hardware products. It describes a set of requirements to be used by cross-functional hardware and software product teams in the selection of the product RAMP attributes.

This standard presents a minimum set of product RAMP requirements; it is not intended to be all-inclusive. It is expected that each product team will identify the RAMP requirements specific to their product.

1.3 RESPONSIBILITIES

1.3.1 Engineering Development Manager

The engineering development manager is responsible for ensuring that the product specification has addressed the product attributes defined by this standard.

1.3.2 Product Team

The product team is responsible for selecting the product RAMP goals and requirements. Each product team member is responsible for providing their organization's product-specific requirements to the product team.

The product team shall select the appropriate set of RAMP attributes required to satisfy the RAMP goals consistent with the product financial constraints. This standard provides a minimum set of RAMP goals and requirements that have to be considered by the product team during this selection process.

The RAMP requirements are worded in two distinct forms to support implementation over a broad range of products.

- The *shall* form of a generic requirement means that the requirement must be addressed for all products. Addressed means that the product team must either incorporate an implementation that satisfies the requirement or provide the justification of why the requirement cannot be implemented.
- The *should* form of a generic requirement means that the requirement may not apply to all products. It is up to the product team to evaluate the applicability of each of the *should* type requirements for their product. Once it is agreed that a *should* type requirement applies, it is then treated as a *shall* type requirement for that product.

After the RAMP goals and requirements have been selected, they are incorporated in the product *Functional Specification*.

1.3.3 Services Representative

The Services representative is responsible for ensuring the service organization requirements address the product RAMP requirements defined by this standard.

1.3.4 Advisory Committee

An Advisory Committee exists for this standard. A current list of members of the Advisory Committee can be obtained from the standard owner.

1.4 RAMP GOALS

The product *shall* have specified RAMP goals that are based on Marketing requirements (customer, competition, market share) and services requirements (revenue, margin contributions, service delivery goals). The product team *shall* define a complete set of RAMP goals based on the Marketing and Services goal inputs. These goals will be used to determine the appropriate level of RAMP requirements for their product.

The underlying assumptions on which these goals are established *shall* be specified (that is, configuration, environment, usage, media).

The design approaches required to achieve each of these goals and the methods required to evaluate the product's performance against these goals before First Customer Ship (FCS) *shall* be specified in the product *Qualification Plan*. All goals stated as *Means, that is, Mean Time to Failure (MTTF), Mean Time to Repair(MTTR)* and others *shall* also specify the type of distribution along with sufficient parameters to uniquely define the distribution.

Refer to Appendix A for definitions for the following set of required RAMP goals.

Reliability Goals

- Inherent product reliability goal
- Observed product reliability goal
- Error rates (when applicable)
- Product fault coverage goal
- Product error recovery goal
- Wear-out life percent survival (when applicable)
- System hardware crash rate (when applicable)
- Inherent field replaceable unit (FRU) reliability goal

Availability Goals

- Inherent product availability goal
- Inherent system availability goal (when applicable)
- Observed system availability goal (when applicable)
- System reboot time goal

Maintainability Goals

- Inherent product maintainability goal
- Product diagnosis goal
- Product FRU fault isolation goals—(one FRU, two FRUs, more than two FRUs)
- FRU removal and replacement goal
- Verification time goal
- Preventative maintenance goal (when applicable)
- FRU Dead on Arrival (DOA) goal

Installation Goals

- Hardware mean time to install goal
- Hardware installation time limit goal
- Software mean time to install goal (if software is installed on the product)
- Hardware problem-free installations
- Software problem-free installations
- Product DOA goal

2 HARDWARE PRODUCT RAMP REQUIREMENTS

2.1 RELIABILITY REQUIREMENTS

All reliability requirements *shall* be formulated and documented. The designer *shall* ensure that the product design provides protection from both internal and external environmental conditions such that the stated reliability goals can be achieved when used under specified operational conditions.

Increasing product reliability through fault avoidance is one of the most effective approaches the designer has for achieving the stated reliability goals. The design engineer *should* use both the selection of the component technology and the selection of the components as one way of achieving the inherent product reliability goal.

It is essential that design disciplines are used that ensure manufacturability and eliminate the possibility of creating manufacturing deficiencies. The product team *should* document the design disciplines they plan to follow to ensure that the parts selected, when implemented in the design, will achieve their individual reliability goals. These disciplines *should* address the following areas:

- Selective screening
- Electrical derating
- Mechanical derating
- Part stress testing
- Noise limits
- Timing analysis
- Module/sub-assembly interaction
- Signal integrity
- Specification compliance
- Packaging
- Mechanical shock
- Thermal integrity

2.1.1 Data Integrity

Data integrity *shall* be maintained throughout the product and over its interfaces. The goal *shall* be no undetected data errors. Design *shall* also strive to minimize the possibility of control errors. The protection styles *should* be matched to the expected failure modes and the expectations of recovery. Systems of data protection *should* overlap throughout the data path so no gaps exist in data protection. There are many error detection and reporting styles from which to choose (that is, many forms of error detection – parity, Error Correction Code (ECC), Cyclic Redundancy Check (CRC), arithmetic codes).

2.1.2 Electrical Design Integrity

Electrical design integrity rules *shall* contain specific derating instructions for electrical loads on the outputs of devices.

The power distribution scheme for the load *shall* be designed so that a fault anywhere in the load circuit will not result in propagation of the failure to exceed the extent of the original failure.

Appropriate values *should* be specified for "ride-through time" and "hold-up time" in order to maintain product operation in its target environment. For example, hold-up times greater than 100 milliseconds may be needed to ride through a majority of power outages for some products.

A capability *should* be provided to prevent the power supply from exceeding current specified limits.

2.1.3 Thermal Design Integrity

Products *shall* meet all aspects of their *Functional Specification*, including reliability performance across the entire range of their environmental specification (Class A, B, or C).

Sufficient cooling capabilities *shall* be provided to allow heat generated within the product to be dissipated out of the cabinet, so the product and component temperatures remain within specification.

The designer *shall* generate a cooling budget for the product that allocates temperature rise across all components. The component layout *shall* be arranged to minimize the impact of heat flow from high heat output devices, and maximize cooling to all components on the module.

2.1.4 Environmental Sensing and Shutdown Protection

Environmental sensing (temperature, supply voltage, current, air flow) *should* be provided. An early warning of approaching limits *should* also be provided as controls to provide for a safe, orderly shutdown when environmental limits are reached to reduce the risk of premature product failure.

All control and status measurement devices *shall* be backed up with limit sensors for critical parameters of power and environment. These sensors should not depend on microprocessors, software, or complex measurement circuits. The sensor(s) *shall* be capable of shutting the product down immediately when triggered; their set points *shall* be sufficiently high that the sensor normally monitoring the parameter will sense an emergency condition before the limit sensor is triggered.

The voltage tolerance, airflow, and temperature detection circuitry *should* be implemented with a means of visual display of state for the customer or service person.

2.2 AVAILABILITY REQUIREMENTS

This subhead addresses fault tolerant availability requirements. All other requirements that support the availability goals are addressed in subheads 2.1 and 2.3 (Reliability and Maintainability, respectively).

The product *shall* be designed to be fault tolerant in support of the overall system Observed Availability Goal. The product *should* be able to recover from detected errors and continue to function normally. The location of error detection mechanisms is key to the ability to recover. Recovery operations *should* be automatic and fully transparent to the customer.

Some of the more effective fault tolerant mechanisms and aids include:

- Retry capabilities
- Error correction capabilities
- Error reporting mechanisms
- Early warning reporting mechanisms
- Use of sanity timers for breaking lockout conditions
- Component and product redundancy
- Configuration management when used with redundancy
- Multiple access paths
- Mapout of fault
- Disabling of a failed component
- Scrubbing (removal) of soft errors
- Bit scattering
- Bit substitution
- Complement/recomplement

Fault tolerant mechanisms *should* be documented. The document *should* include a description of all error indicators and signaling mechanisms. The document *should* define all possible error conditions and error handling algorithms to be used to recover from error conditions.

2.3 MAINTAINABILITY REQUIREMENTS

This subhead describes the hardware product maintainability requirements.

2.3.1 Product Requirements

The product *shall* be designed to satisfy the stated maintainability goals. The *Mean Time to Repair* (MTTR) goal is broken out into the components of a typical service call:

1. MTTD—*Mean Time to Diagnose and Isolate*
2. MTTRR—*Mean Time to Remove and Replace*
3. MTTVR—*Mean Time to Verify Return* to normal operations

2.3.1.1 Error Detection

Effective error detection is a necessary prerequisite for all forms of data integrity, error recovery, and fault isolation. Detection mechanisms *shall* be provided in support of reliability (data integrity), availability (dynamic error detection/recovery), and maintainability goals (error detection and quick isolation). Error detection mechanisms *shall* be located as close as possible (in time and distance) to the logic being covered. This *shall* extend down to the fault coverage within a chip, when applicable. Product FRUs *shall* be designed for testability and repairability. The product *shall* include sufficient detection mechanisms to meet its fault coverage and MTTD goals. The amount of fault coverage *shall* be based on the availability, recovery, and fault isolation goals.

The product design engineer, for example, *should* provide a means for detecting missing clocks, ac and dc input power status, environmental conditions outside of tolerance thresholds, and time-out errors on busses and data paths.

Some products will require the use of a *Watchdog Timer* (Keep Alive) mechanism to ascertain that an error has occurred within the product or at an interface and the product is unable to communicate the error condition.

Wear-out detection *shall* be provided for those items that have an expected replacement time less than the expected life of the product. Detection and scheduled replacement *shall* occur before the product's availability to the customer is impacted. (For example, replacement based on the specified life expectancy of a tape head, the thickness of wear bands.)

A viable means *shall* exist to test every piece of error detection logic to verify that it is functioning correctly (that is, check the checkers). Using a programmable means of testing is preferable to a physical means. Use of software fault insertion techniques to activate error detection circuits is a good example of an effective way to dynamically test error/event logging, error handling, and diagnostic tools.

2.3.1.2 Error Information Capture and Reporting

The detection logic for a given error condition *should* record all pertinent symptom information at the time of detection and initiate the recovery and reporting mechanisms.

The information available for capture *shall* contain enough error and state information to identify the failing FRU, to support error recovery, and design/process troubleshooting. For example, silo registers are often used to monitor bus activity, record status, save instruction counters, and parity check information of the most recent cycles of bus or processor activity. Registers are commonly used to capture machine states, usage, and status information.

Early warning operational indicators *should* be provided in a product for any conditions that can be either automatically or manually corrected before operations are impacted.

In the event of a secondary error condition, the design *should* provide a means of retaining the original symptom information so it will not be overwritten by the secondary error.

All fault and status condition indicators within the power complex that can be communicated *should* be software readable. Mechanical latching indicators *should* be used for catastrophic failures.

2.3.1.3 Fault Isolation

The design *shall* provide error detection mechanisms close enough to the source of a fault along with sufficient related symptom information to support FRU isolation.

Using computer-aided design (CAD) simulation tools in the support of the layout and evaluation of the effectiveness of the placement of detectors in support of the isolation goals is recommended when feasible.

The effectiveness of fault isolation capabilities of test directed diagnostics, symptom directed diagnosis, and troubleshooting flows/maps all depend heavily on the placement of the detectors designed into the product.

Basic troubleshooting aids *shall* be provided for isolating the more difficult faults that are not isolated by the product or the diagnostic set. The ability to run voltage and clock margins is required, where appropriate, to ensure that intermittent symptoms can be replicated for diagnostic tool analysis and isolation. Margining of power supplies will only be executed on whole power system segments; multiple regulators *should* track together.

2.3.1.4 On-site Repair Process—Physical Repair

The product design *shall* be as resilient to human error as possible in support of product maintenance. The design *shall* support each of the following goals associated with performing the repair.

FRU Access, Removal and Replacement

The design *shall* provide the capability necessary for the removal and replacement of any FRU within the specified FRU Mean Time To Remove and Replace (MTTRR) goal. This goal covers the time from the physical removal of the product's outer covers to the replacement of these covers. Packaging and component layout *shall* allow for quick access to all FRUs for removal and replacement within the stated timing goals by either Customer Services or the customer. All FRUs *should* be identifiable by serial number, part number, and revision level, without FRU removal. A clear and concise chart of all FRUs *shall* be provided at the normal service access point within the cabinet.

Design *shall* support replacement of all FRUs by one person. Each FRU *shall* be replaceable without requiring the removal of other FRUs. It *should* be possible to remove any FRU without the use of tools or the disconnection of any signal interconnects, unless directly connected to the FRU being replaced.

No cables *should* exist between modules (or other common FRUs) that impede removal and replacement of an FRU or require removal of other FRUs.

FRUs *should* be either physically or electrically keyed to ensure proper replacement. If electrically keyed, the design *shall* ensure that no damage can occur if the module is accidentally placed in a wrong location. All FRU components (MCUs, EEPROMS, DRAMs, cables) *shall* be labeled or keyed to ensure proper replacement. When used, mounting hardware *should* be captive to the FRU or the mounting. Captive, quick release fasteners are preferred for retaining modules and access covers. Static protection capabilities *shall* be provided for handling FRUs that contain electrostatic-sensitive devices.

Regulators *should* be pluggable, not hard-wired to the power supply assembly.

Online Maintenance

The ability to individually power on/off and deselect a product or FRU that needs replacement while the remainder of the system continues to operate (for example, use of pluggable storage devices can support a warm swap repair strategy) *should* be provided. Many components cannot support warm or hot swap unless a sufficient amount of redundancy, backup, and/or shadowing is also provided. Powering off/on any product *should not* impact customer operations.

A test capability *should* be provided that will verify functional operation and correct installation prior to reconfiguring the product back onto an operating system, after it has been repaired using either a warm or hot swap of an FRU.

Adjustments and Jumpers

Configuration switches or jumpers *shall not* be required to perform field maintenance. The configuration function *shall* be *programmable*.

Field adjustments to the product *shall* be minimized, both for preventive maintenance and for corrective maintenance. The goal to strive for is the elimination of field adjustments. Supply voltage or internal monitor adjustments *should not* be required for service.

Any necessary adjustments *should* be supported by software tools that can be used both locally and remotely. Adjustments *should not* require the use of meters, scopes, or special equipment.

When required, manual adjustments *should* be easily accessible, done with a minimum of subjectivity, provide a positive success indication, and induce no interaction effects. The adjustments *should* be zero center and clockwise increase, fine adjustments by larger movement, protection against accidental movement, automatic compensation for drift and wear, and control limits. Access to any adjustments *shall not* require the use of module extender boards in the field.

Electrical and Mechanical Tools

Only those common tools supplied in the services tool kits *should* be necessary to service or install the product. Services *shall* provide the project team with a list of common tools that will be included in the tool kit that supports the product. To minimize the cost of service, the use of special tools (for example, torque wrench, special tester), must be cost-justifiable.

If special electronic tools are required, the necessary connection mechanisms (such as, adequate power, properly grounded receptacle) *shall* be provided. A power outlet with a system common ground *shall* be provided within the cabinet to ensure accurate test equipment measurements without ground loop interference.

Connectors and Cables

The requirement is to reduce the possibility of a misconnection by minimizing the number of cables or cable assemblies used in the design. Where required, cables *shall* be labeled, strain relieved, accessible in all possible installations, and located so as to prevent inadvertent disturbance while performing service on the cabled device or adjacent devices. The corresponding receptacles *shall* also be clearly labeled.

All connectors *shall* be keyed, have positive interlocking, quick disconnect, and permit interconnection only with the corresponding plug or connector. The need for hand tools *should* be minimized.

A concise table of cable routing and part numbers *shall* be provided within each cabinet.

2.3.1.5 Verify/Restore

The product team *shall* develop verification procedures and support tools that will support the achievement of the products Mean Time To Verify and Restore (MTTVR) goal. The verification procedure includes the following steps:

1. Verify the new FRU or repair is indeed functional.
2. Clean-up by setting system back to the original state.
3. Verify the system is operational.
4. Verify the original fault has been removed.

2.3.1.6 Revision and Control

The requirement is to provide machine-readable identification that includes the part number, revision level, and serial number for all products and FRUs. This information *shall* be accessible to higher level entities, the operating system, console, and network managers.

2.3.1.7 Safety

All products covered by this standard *shall* meet Digital safety requirements.

Dangerous exposed voltages must not remain in removed FRUs. Any required capacitor bleed-down time *shall* be clearly marked on the FRU faceplate.

Emergency ac power shutdown *shall* be executable by tripping an external total OFF switch or circuit breaker at the normal service entry point within the cabinet.

2.3.2 Diagnosis Requirements

The overall product diagnosis strategy *shall* support product fault coverage, diagnosis, isolation, and run-time goals. Ideally, the combination of fault coverage and the application of diagnostic tools *should* provide 100% isolation to a single field replaceable unit within the specified diagnosis time goal. The product Diagnostic Engineer(s) along with the product team *shall* define the detection scheme, diagnosis approach, and supporting diagnosis features. The challenge is to select the right combination of detection and diagnosis capabilities to achieve the maintainability goals. The diagnosis approach *shall* document how the various diagnosis techniques will be used to achieve the overall maintainability goals. The error detection mechanisms and related state information required to support the diagnostics goals *shall* be specified. Fault detection coverage *should* address those structures that are most likely to fail and *should* cover near 100% of the data paths critical to higher level test, error detection circuits, data arrays (for example, caches, memory, translation buffer) and element interconnects.

Test directed diagnosis (TDD) and symptom directed diagnosis (SDD) are examples of primary diagnosis strategies. Normally a combination of test directed diagnosis and symptom directed diagnosis tools *should* be developed to achieve the maintainability goals.

2.3.2.1 Symptom-Directed Diagnosis Requirements

Symptom-directed diagnostics *shall* be designed to determine the cause of an error directly from the evidence or symptoms available at the time the error was detected. The error detection mechanisms and the related state information required to support the fault isolation goals *shall* be specified.

2.3.2.2 Test-Directed Diagnosis Requirements

Test-directed diagnostics *should* be packaged to execute as either built-in tests, off-line loadable tests, on-line loadable diagnostics, or a combination of all three.

The built-in tests are preferred implementations of test-directed diagnostics because they can be automatically invoked (ease of use), provide a high confidence in the results (better support MTTR goals), provide minimal test time, and provide ease of maintaining revision levels. These tests *should* be implemented at each physical level within the system. Testing levels *should* include:

- Mechanical interconnects
- Basic chip functionality
- FRU (module-based)
- Element (host, storage, communications)
- Cabinet and interconnect

Test-directed diagnostics *should* test basic functionality first. After a piece of logic has been tested and found to be error-free, it is assumed to be error-free in subsequent tests.

Off-line tests may be required if the product is non-boot critical and the built-in tests are not feasible or they can not provide full coverage. Off-line diagnostics are functional tests that *should* be capable of executing either free standing or under a diagnostic supervisor.

On-line diagnostics *should* be designed to complement the built-in tests. They are functional tests that execute under an operating system and are used primarily to provide fault coverage over product interfaces and interactive situations beyond the coverage provided by built-in tests.

Diagnostic Selection and Initiation

Built-in tests *should* be capable of initiation on power-up and on command from higher level functions, including the console. These tests *should* be capable of running without user intervention. The user *shall* be able to access the test results and also be able to change any run-time parameters.

The entire on-line and off-line diagnostic selection and input parameter process *should* be as automated as possible (that is, auto configure, selection of default parameters). Any required input parameters *shall* be easily understood requests (present clearly understood input alternatives); defaults are required for all inputs (and should be the preferred or most likely input). A capability *should* be provided to save and easily modify customized setups to support quicker verifications for subsequent runs. Checksum/verify logic *should* be included in all loadable diagnostic programs.

Remote diagnostic capabilities *shall* include downline loading of files and executables, full console functionality, remote dial-in, verification, and authentication. All remotely run on-line diagnostic activities *shall* be capable of being logged to the event/error log.

User Protection From Diagnostic Tools

The diagnostics *shall* be designed to preclude inadvertent corruption of system or user data. The diagnostics *shall* determine if adequate load and execution area is available for loading and running diagnostics (for example, make provisions for reserved storage and access areas, verify basic functionality before executing diagnostics, inform user of any potential risks to the system).

Diagnostics *should* automatically verify that the product under test is at the proper revision level and is properly configured.

Diagnostic Execution

Diagnostics *should* be designed to continue the diagnosis process whenever the failing FRU cannot be identified based on the original symptom. The diagnostics *should* be capable of executing additional test sequences until sufficient symptom information is available to identify the FRU. The diagnostics *should* provide appropriate troubleshooting support, like the options to stop/continue/loop on error.

Diagnostics *should* be designed to prevent hanging or halt during execution (hangs are not considered detected errors in support of service personnel). Some form of time-out mechanism *should* be employed.

Built-in loopback connectors (preferred implementation) or external loopback connectors *shall* be supported by the diagnostics.

Diagnostics *should* be capable of operating from any CPU within a multiple-CPU environment.

Diagnostic Reporting

Diagnostics *shall* report detected errors in an easily understood fashion that directs the user to a repair action. All loadable diagnostic tools *shall* identify themselves, including the revision level, on all error messages. All on-line diagnostics *shall* be capable of logging errors to the event/error log. During execution of a diagnostic program, the current progress of the test relative to the total expected run-time *should* be displayed graphically. The display *should* be capable of displaying run-time parameters for all diagnostic programs running simultaneously.

2.3.2.3 Exerciser Program Requirements

Exerciser program requirements *shall* be specified. The following when applicable, *should* be considered:

- Exercise the entire system.
- Require minimal user input.
- Automatically size the system and set up a load appropriate for the system under test.
- Provide the user with the option of overriding operational parameters (components to be exercised, number of iterations, size of load area).
- Verify results for each exercised component.

2.3.2.4 Utility Program Requirements

Utility program requirements *shall* be specified. The following when applicable, *should* be considered:

- Special product analyzers
- Error/event log summaries
- Crash dump analyzers
- Special product support functions (formatters and adjustment support)
- Remote diagnostic utilities

2.3.2.5 Verification of Diagnostic Tools

The quality and effectiveness of the diagnostic tools *shall* be verified through either the use of simulation or physical fault insertion techniques or a combination of both. Full regression testing *shall* be provided to support diagnostic checkout and updates.

2.3.3 Training Requirements

Product training courses *shall* be provided to support service personnel in the achievement of the product service goals. The training courses *shall* encompass all aspects of operating, installing, and servicing the product. The prerequisites for each training course *shall* be defined.

Product training courses *should* be designed to meet the service delivery needs of three primary audiences, at a minimum.

1. The first audience consists of process-focused (first level) *service engineers*. Training for this audience *should* cover procedures for performing product installations and maintenance. The training *should* focus on the product service strategy, product operational characteristics, fault isolation tools and techniques, hands-on use of the related service tools and documentation, and electrical safety practices including any training required by a regulatory agency (for example, the Occupational Safety and Health Administration (OSHA) in the United States).

2. The second audience consists of *remote diagnosis engineers*. Training for this audience *should* focus on fault isolation techniques and tools. The training relies heavily on the student possessing a basic understanding of system software and architectures. This audience most often resides in centralized diagnosis centers.
3. The third audience consists of *product support engineers*. This audience requires an in-depth understanding of product theory of operations. The training relies heavily on the student having an in-depth knowledge of operating system environments, system architectures, networks, and electrical safety practices including any training required by a regulatory agency (for example, the Occupational Safety and Health Administration (OSHA) in the United States).

The training plan for each product will normally address the needs of the previous audiences, although a course may not be required for each.

Engineering Development *should* provide a technical seminar before First Revenue Ship (FRS) that addresses all aspects of the product, especially any unique areas of functionality.

2.3.4 Documentation Requirements

Product documentation *shall* be provided to support users in the achievement of the product service goals. The documentation *shall* encompass service personnel needs and the service-related needs of our customers, where applicable. The documentation set *shall* also be available for electronic distribution.

2.4 INSTALLATION REQUIREMENTS

The installation goals are to achieve 100% problem-free installations (PFIs), to complete the installation within the specified MTTI goal, and to ensure that the customer's perception will be that the products and service are of the highest quality.

Both the product and its shipping container *should* be designed to ensure the product will function properly when installed at the customer site.

Design *shall* allow installation without the need for disassembly.

Unpacking instructions *shall* be printed on or attached to the outside of each shipping container. The container *shall* be clearly marked and each component easily identified for installation purposes. The use of international symbols, pictures, and photographs are recommended. It *shall* be clear which container is to be opened first. The *Installation Guide* and software test procedures (scripts) *shall* be shipped with the product and *shall* be readily available/accessible when the shipping container is opened.

The initial product tests *should* be capable of running independent of an operating system. Product installation documentation and Customer Services training are key contributors to the achievement of the installation timing goals.

3 SOFTWARE AND FIRMWARE SUPPORT OF HARDWARE RAMP REQUIREMENTS

All software and firmware components that support a given set of hardware products *shall* support the specified RAMP goals of those hardware products. Refer to Appendix A for definitions of the product RAMP goals.

3.1 RELIABILITY REQUIREMENTS

Operating systems *shall* provide product data integrity protection against spurious or inadvertent faulty accesses by using techniques such as consistency and range checking.

3.2 AVAILABILITY REQUIREMENTS

The availability requirement is to continually satisfy the customer's essential computing demands by providing the ability to recover from faults which either temporarily or partially degrade system performance. Fault tolerance, covers such areas as error correction, error retry/recovery and redundancy, can increase the perceived reliability and availability of products.

3.2.1 Configuration Monitoring

System software and firmware *shall* support system configuration management by making use of hardware product RAMP identification features. Systems and networks *shall* automatically maintain up-to-date configuration files in support of effective system fault management.

On initiation, programs *should* be able to automatically tailor their execution, based on information gathered through machine-readable parameters; any nonconforming configurations *should* be flagged and correct configurations presented to the user. Model numbers, serial numbers, revision levels, operational or execution times, and other product-specific parameters (both hardware and software) *shall* be supported by the system. A capability *shall* exist to manually override any automatically generated configuration.

3.2.2 Product Error Handling and Reporting

Software/firmware facilities *shall* be provided that support error handling and reporting. The software/firmware used to support new hardware products *shall* make all efforts to recover from an error condition. The software/firmware *shall*:

- Ensure that errors are detected and contained within the failing software module.
- Ensure that hardware-detected errors will not result in a system crash or premature termination of an application, unless the cause of the error could adversely affect system or data integrity.
- On the detection of an error, use an operating system error handling facility to manage the hardware and software exceptions.
- Provide for automatic system restart that returns the system to an operating condition after a power interruption or fatal error. All products *should* be designed to support auto-restart and to log the event to the error/event log.

3.2.3 Fault Tolerance Support

One method of achieving fault tolerance is to provide redundancy for hardware resources critical to system operation. Systems and networks *shall* provide software/firmware support for redundancy in hardware products, data paths, and critical data so no subsystem is a single point of failure.

The effect of faults on the system *shall* be minimized. If resources or entities of the system must be removed to bypass a fault, only the smallest part of a resource that can be deconfigured and fully contains the fault *should* be deconfigured, or the smallest entity affected by the fault *should* be removed. Examples include moving single pages of memory to a bad page list or forcing only the current image or process to exit instead of rebooting the system. The system *shall* provide a means of returning a resource to the system after that resource has been repaired.

A nonrecoverable error or a recoverable error that occurs above a specified threshold rate *should* result in the task being rescheduled to another resource (when available) and that the unit that reported the error should be withdrawn from use. A capability *shall* exist to manually override the results of automatic reconfiguration. Systems *should* automatically keep track of removed resources to avoid reconfiguring these resources back into the system at the next system initialization. All configuration changes or resource removals *shall* be logged and the operator notified.

The software/firmware *shall* support product fault tolerant features (for example, support for alternate access capabilities, switching off of cache, and computer-initiated failover).

3.3 MAINTAINABILITY (SERVICEABILITY) REQUIREMENTS

The product *should* be designed for remote maintenance, up to the point of physical repair. The strategy is to be able to remotely diagnose a fault to an FRU and provide a solution to the customer in the form of a workaround or a fix that will result in a scheduled/deferred maintenance call or a permanent fix by means of a remote patch to software or firmware. Firmware updates *shall* be down-line loadable.

Each hardware product's specification provides detailed information about the maintenance support required of the software/firmware. This information will include the error reporting mechanism, error recovery procedure, information to be logged, translation of product register/status information, and information to be used by the fault diagnosis and isolation tools.

3.3.1 Diagnostic Support

Software/firmware *shall* support product built-in and on-line diagnostics. Examples of this type of support include initiating and reading results of ROM-based diagnostics, loopback and turnaround of data, and remote access.

All configuration settings, revision levels, entity status indications, and physical switch settings *should* be ascertainable both locally and remotely.

3.3.2 Event/Error Logging Support

Event/error logging *shall* be an integral part of the system. Identify, for each error, the hardware and software state information to be logged with the entry.

3.3.3 Revision Management

The operating system *should* ensure the revision levels for high-level assemblies are at their proper level at power-on and boot time. When they are not, a message *shall* be sent to the console and to the error/event log. Product software and firmware *shall* support this system-level revision control.

The operating system *should* be capable of retaining the network address across FRU replacement. Revision and ECO level of all network entities *should* be ascertainable by means of the network.

3.4 INSTALLATION REQUIREMENTS

Software installation procedures and verification programs *should* be as automated and simple to execute as possible. To successfully install the product, the procedures *should* require only limited system knowledge and *should* minimize the probability of failure. The installer *should* only be required to know specific site information to install the product. Whenever possible, the system *should* be shipped prebuilt (factory preload of all software).

Product installations *shall* use the utilities provided by the operating system.

3.4.1 Software Installation Procedures

The software installation procedure *shall* keep the installer informed of what is happening. This information is needed both as an aid in troubleshooting any problems and for informing the installer that the installation is indeed proceeding.

The installation procedure *shall* specify an estimated time for the installation. Prior to commencing the actual installation, the installation procedure *should* check for required disk space, other needed resources, and required revision levels .

The default set-up *should* be as secure as possible for the operating system and layered accounts. All accounts set up during product installation *shall* have passwords; these passwords *shall* be generated and changed during the installation process.

The installation procedure *should* allow the user to answer all installation questions at the beginning of the installation; the installer would then not have to be present during the remainder of the installation. Default parameters *should* be displayed and help information available for all input prompts. Installers *should* be able to modify product characteristics during the installation without being required to restart the installation process from the beginning.

Any user command line interface *should* provide a command syntax consistent with the operating system's syntax. All commands must be registered with the appropriate operating system if applicable.

3.4.2 Verification Tools

An Installation Verification Procedure (IVP) *shall* exist. It *should* be based on IVP utilities provided by the operating system and provide a complete check of the product's operability.

3.5 SOFTWARE PRODUCT DOCUMENTATION

This subhead describes requirements for software product documentation.

3.5.1 Software Installation Manual

Software Installation Manuals *shall* provide the following information:

- List of options that are part of the product
  - — Manuals
  - — Distribution media
  - — Cover letter
- List of files generated or affected by the product installation
- List of accounts generated or affected by the product installation
- Installation prerequisites
  - — Hardware (amount of memory, required options, disk space, revision control)
  - — Software (products/revisions)
  - — System resources (paged pool, nonpaged pool)
  - — User parameters (privileges, quota)
  - — Product naming conventions (product names, system name for product type)
- Worksheets
  - — Answers to all installation questions asked
  - — Estimated installation time per program
- A sample installation script
- Release notes
  - — Changes to product software documentation
  - — Changes from previous version
  - — New features
  - — New hardware supported
  - — Known restrictions
  - — Bug fixes
  - — Compliance to standards
  - — New error messages

3.5.2 Software Documentation

The software product team *shall* specify the content and the intended audience of each of the following software documents:

- User's guide
- Reference manual
- On-line help documentation

APPENDIX A EXPANDED DEFINITION OF RAMP GOALS

RELIABILITY GOALS

- *Inherent Product Reliability Goal*

  Inherent product reliability is the potential reliability of an item present in its design. Reliability is defined as the probability that an item will perform its intended function without failure for a stated period of time under stated conditions.

  Inherent product reliability is calculated as the Mean Time Between Failures (MTBF) in hours. MTBF is the arithmetic average time interval between failures of an operating product. MTBF is computed as the ratio of the cumulative time to the number of failures under stated conditions. Only product hardware faults that caused the product not to perform its intended function are counted as a failure (and does not include the use of fault tolerance within the design).

- *Observed Product Reliability Goal*

  Observed product reliability is the reliability of an item that can be achieved above that of inherent reliability when fault tolerance is included in the product design.

  Observed product reliability is calculated as the Mean Time Between Failures (MTBF) in hours. MTBF is the arithmetic average time interval between failures of an operating product. MTBF is computed as the ratio of the cumulative time to the number of failures under stated conditions. Only those product hardware failures, after fault tolerance has been invoked, that caused the product not to perform its intended function are counted as a failure.

- *Error Rates* (when applicable)

  Error rate is a measure of the quality of a circuit or system; it is expressed as a ratio of the number of erroneous bits or characters, frequently taken per 100,000 characters, to the total number of bits or characters in a sample. It is most often used to define acceptable operational thresholds for recoverable environmental- or media-induced faults.

  The error rate is calculated as the ratio of the number of errors per unit of life (time, cycles, characters printed, seeks, transmits and receives) to the total number of units in a sample.

- *Product Fault Coverage Goal*

  The product fault coverage goal is defined and calculated as the percentage of the product's total failure rate covered by error detection mechanisms.

- *Product Error Recovery Goal*

  Product error recovery is defined and calculated as the percentage of detected errors that invoke recovery operations (and includes the use of fault tolerance in the design).

- *Wear-Out Life Percent Survival* (when applicable)

  Percent survival is the percentage of items that continue to function beyond a specified point in the product's life. The 90% survival point shall be specified for a product whose failure rate is not constant or for products whose expected life is less than the useful life of the product. It is recommended that the percent survival be specified at two points in the product's life to define the product's wearout characteristics more precisely. The units of product life used in the percent survival will depend on the nature of the product. Examples are hours, cycles, and characters printed.

  Wear-out life percent survival:

  ```
  90% at ____   and   ___% at ____   for each applicable part #.
  ```

- *System Hardware Crash Rate* (when applicable)

  The system hardware crash rate is made up of all crashes resulting from product hardware faults. A crash is defined as an event that results in the simultaneous disruption of service and loss of work on all components of the system.

  The system hardware crash rate is calculated as the arithmetic average of the number of system hardware crashes per year attributed to this product.

- *Inherent Field Replaceable Unit Reliability Goal*

  The same reliability definitions used for Inherent Product Reliability apply to this goal.

  Inherent Field Replaceable Unit reliability is calculated as the Mean Time Between Failures (MTBF) of an FRU in hours. MTBF is the arithmetic average time interval between failures of an operating FRU. MTBF is computed as the ratio of the cumulative time to the number of failures under stated conditions. Only those FRU hardware faults that caused the FRU to not perform its intended function are counted as a failure (and does not include the use of fault tolerance within the design).

AVAILABILITY GOALS

- *Inherent Product Availability Goal*

  Inherent product availability is the ability of a product, under combined aspects of its inherent reliability and maintainability support, to perform its required function at a stated instant in time. The time considered is operating time and active repair time and does not include the use of fault tolerance mechanisms. Availability is defined as the probability that an item will be operational at an instantaneous point in time when used under stated conditions.

  Inherent product availability is calculated as a percent based on the equation [MTBF/(MTBF + MTTR)], where the MTBFs in the equation are the arithmetic average time interval (in hours) that may be expected (not using fault tolerant mechanisms) between failures of an operating product. The MTTR in the equation is the arithmetic average active time required to repair a defective product. The active repair time only includes the time to diagnose (MTTD), effect a fix (MTTRR), and perform checkout (MTTVR).

- *Inherent System Availability Goal* (when applicable)

  Inherent system availability is the ability of a product, under combined aspects of its inherent reliability and maintainability support, to perform its required function at a stated instant in time. The time considered is operating time and active repair time and does not include the use of fault tolerance mechanisms. Availability is defined as the probability that an item will be operational at an instantaneous point in time when used under stated conditions.

  Inherent system availability is calculated as a percentage based on the equation [MTBF/(MTBF + MTTR)], where the MTBFs in the equation are the arithmetic average time interval (in hours) that may be expected (not using fault tolerant mechanisms) between failures of an operating system. The MTTR in the equation is the arithmetic average active time required to repair a defective system. The active repair time only includes the time to diagnose (MTTD), effect a fix (MTTRR), and perform checkout (MTTVR).

- *Observed System Availability Goal* (when applicable)

  Observed system availability is the ability of a product, under combined aspects of its observed reliability and maintainability support, to perform its required function at a stated instant of time. The time considered is operating time and active repair time and includes the use of fault tolerance mechanisms. Availability is defined as the probability that an item will be operational at an instantaneous point in time when used under stated conditions.

  Observed system availability is calculated as a percent based on the equation [MTBF/(MTBF + MTTR)], where the MTBFs in the equation are the arithmetic average time interval (in hours) that may be expected (using fault tolerant mechanisms) between failures of an operating system. The MTTR in the equation is the arithmetic average active time required to repair a defective system. The active repair time only includes the time to diagnose (MTTD), effect a fix (MTTRR), and perform checkout (MTTVR).

- *System Reboot Time Goal*

  System reboot time is defined as the amount of time to execute a reinitialization (reload) of the system.

  System reboot time is calculated as the mean time to reboot the system after the system has been taken down in support of a repair activity. Time is calculated in minutes.

MAINTAINABILITY GOALS

- *Inherent Product Maintainability Goal*

  Inherent product maintainability is defined as that portion of maintainability built into the product. Maintainability is defined as the ease with which the maintenance of a product can be performed in accordance with prescribed requirements.

  Inherent product maintainability is calculated as the active Mean Time to Repair (MTTR) a product, in hours. It is restricted to active time to diagnosis (MTTD), effect a fix (MTTRR), and perform a verification (MTTVR).

- *Product Diagnosis Goal*

  Product diagnosis is defined as the Mean Time to Diagnose (MTTD) the location of the fault and to determine the required repair action.

  MTTD is calculated as the arithmetic average time to isolate the cause and location of the fault.

- *Product Fault Isolation Goals*—(one FRU, two FRUs, more than two FRUs)

  Product fault isolation is defined as the probability of isolating a fault to one, two, or more than two FRUs.

  Product fault isolation is calculated as the percentage of time a fault can be isolated to:

  — A single FRU (number of times a single FRU called out/total number of callouts)

  — Two FRUs (number of times two FRUs called out/total number of callouts)

  — More than two FRUs (number of time three or more FRUs called out/total number of callouts)

- *FRU Removal and Replacement Goal*

  The FRU removal and replacement parameter is defined and calculated as the Mean Time to Remove and Replace (MTTRR) an FRU. MTTRR is the arithmetic average time to gain access to, remove, and replace an FRU within the product.

- *Verification Time Goal*

  The verification time parameter is defined and calculated as the Mean Time to Verify the Repair (MTTVR). MTTVR is the arithmetic average time to verify that:

  — Replacement part or parts are operational

  — Original error condition has been removed

  — Product is available to the system

- *Preventative Maintenance Goal* (when applicable)

  Preventative maintenance is defined as the maintenance carried out at predetermined intervals or corresponding to prescribed criteria, and intended to reduce the probability of failure or the performance degradation of an item.

  Preventative maintenance is measured as the Mean Time Between Preventative Maintenance (MTBPM). MTBPM is the arithmetic average time between preventative maintenance activities.

- *FRU DOA Goal*

  DOA is defined as those products or components that did not exhibit expected behavior when first installed.

  FRU DOA is calculated as the percentage of FRUs that were dead on arrival relative to all FRUs installed. The goal to strive for is 0% DOAs.

INSTALLATION GOALS

- *Hardware Mean Time to Install Goal*

  Hardware Mean Time to Install (MTTI) is defined as the mean time to unpack, set up, connect, and verify the hardware product is operational. It does not include administrative, travel, or waiting time.

  The hardware MTTI is calculated as the arithmetic average time to perform the hardware product installation.

- *Hardware Installation Time Limit Goal*

  The hardware installation time limit is defined as a threshold for measuring the installation times that tend to be log-normal, implying a *tail of the dragon* phenomenon.

  The hardware installation time limit is calculated as the percentage of hardware product installations completed within a stated time limit.

- *Software Mean Time to Install Goal* (if software is installed on the hardware product)

  Software Mean Time to Install (MTTI) is defined as the mean time to set up, load, and check out that the product is operational. It does not include administrative or travel time.

  The software MTTI is calculated as the arithmetic average time to perform the installation of the software product.

- *Hardware Problem-Free Installations*

  A problem-free installation (PFI) is defined as a product installation for which no problems were recorded. An installation problem, as defined here, is any hardware event that prevents the installation from being completed.

  Hardware problem-free installations are calculated as the percentage of hardware product installations that were problem-free relative to the total number of hardware product installations. The goal to strive for is 100% problem-free installations.

- *Software Problem-Free Installations*

  A problem-free installation (PFI) is defined as a product installation for which no problems were recorded. An installation problem, as defined here, is any software event that prevents the installation from being completed.

  Software problem-free installations are calculated as the percentage of software product installations that were problem-free relative to the total number of software product installations. The goal to strive for is 100% problem-free installations.

- *Product DOA Goal*

  Product DOA is defined as those products that did not exhibit expected behavior when first installed.

  DOA is calculated as the percentage of products dead on arrival, relative to all products installed. The goal to strive for is 0% DOAs.

GLOSSARY

BIST: Built-In Self-Test. A procedure contained within, and processed by, one or more circuits, modules, subsystems, or systems for the purpose of evaluating its host environment.

Data Integrity: The quality of data that exists, as long as accidental or intentional malicious destruction, alteration, or loss of data does not occur.

Default: Pertaining to an attribute, value, or option assumed when none is explicitly specified.

Derating: The intentional reduction of stress/strength ratio in the application of an item, usually for the purpose of reducing the occurrence of stress-related failures.

Device: A mechanical, electrical, or electronic contrivance with a specific purpose.

Diagnose: To systemaically determine the cause of a problem, based on the analysis of abnormal behavior.

Diagnosis: A process of discerning abnormal behavior and systematically determining its cause.

Diagnostic: Any tool, procedure, or activity used to detect and isolate malfunctions.

DOA: An acronym for Dead On Arrival. It is used to describe either a product or a component that did not exhibit expected behavior when first installed.

Error: Any discrepancy between a computed, observed, or measured quantity and the true, specified, or theoretically correct value or condition.

Error/Event Log: A data set or file in a product or system where error or event information is stored for later access.

Error Handling: A process of handling a single error event by attempting recovery.

Error Handling Specification: A product design specification that contains the details of error handling algorithms for a specific product.

Error Recovery: A process of correcting or bypassing the effects of a fault to restore a computer system to a prescribed condition.

Event: A significant occurrence or happening.

Failure: The termination of the ability of a functional unit to perform its required function. A failure is the result of a fault.

Fault: Any condition that causes a unit to fail to perform its required function. It is considered the underlying cause of an error, although it may or may not be physically identifiable.

Fault Management: The recognition of a fault condition and prescription of corrective action based upon a diagnosis.

Fault Tolerance: The ability of a system to deliver proper (expected) service after faults have manifested themselves.

FRU: Field Replaceable Unit. A unit of a system's structure that can be readily removed and replaced at the customer site.

Functional Test: An evaluation that presents sufficient stimuli to cause a device to exhibit specified functionality and measures that which is evidenced against that which is specified.

Installation Verification Procedure (IVP): A procedure distributed with the system that verifies that basic facilities are functioning properly.

Intermittent: Failure of an item for a limited period of time, following which the item recovers its ability to perform its required function without being subjected to any external corrective action.

Isolation: A process used, following error detection, to gather and analyze symptom data in order to localize the failing component.

Maintenance: Any activity intended to retain a functional unit (equipment or program) in, or to restore it to, a state in which it can perform its required function.

Module: Generally, a separable logic component. In hardware, this normally equates to a pluggable entity: a device, chip-carrier, board. In firmware and software, this refers to a loosely connected, functionally defined code block.

Online: Pertaining to the operation of a functional unit when under the direct control of an operating system.

Predictive Diagnosis: A process that attempts diagnosis based on observation of operating conditions over time.

RAMP: Reliability, Availability, Maintainability Program. The collection of goals, strategies, and requirements specified to ensure product success and profitability in the marketplace from a service viewpoint.

Recoverable Error: An error condition that allows continued execution of a program and does not lead to a failure.

Repair: The removal of a fault from the system. The fault may be physical (such as FRU replacement, software patch) or logical (bad page mapout). Repair also includes the concepts of repair clean-up, and temporary and permanent repair.

SDD: Symptom Directed Diagnosis. A diagnostic strategy that seeks to determine cause of failure directly from the symptoms exhibited.

Self-Test: A process within an entity that validates its own functionality.

Standalone: Pertaining to the operation of a functional unit when not under the direct control of an operating system.

System: An interacting set of subsystems that provide a service to the user. Note that the subsystems can be further subdivided to the desired level of detail. In data processing, a collection of people, machines, and methods organized to accomplish a set of specific functions.

TDD: Test Directed Diagnosis. A diagnostic strategy that seeks to determine the cause of failure from the results produced by a battery of *test cases*.

Test: The operation of a functional unit and comparison of its achieved result with the defined result to establish acceptability.

Validation: The checking of a design or data for correctness or compliance with applicable standards, rules, and conventions.

Verification: The act of determining whether a design or an operation has been accomplished correctly or has satisfied its specification.

Watchdog Timer: A device used to discern whether a prescribed condition is present within a predetermined time period (often referred to as a *Keep Alive* function).

Wear-out Failure: Failure due to normal deterioration or mechanical wear, the probability of which increases with usage.